TS Bulletin #1-91

Published As A Hobby By:
BILL HARMER
97 Ruskin Street
Ottawa, Ont. K1Y 4B3
CANADA

Supporting low cost computing & computer literacy since 1987.//Backissue Price $3

--TS Bulletin Reg.#1-1991 - © in the public domain - July 18,'91 -Photocopy It!-8 pp.-
ver. 9.9

Here it is, nearly the middle of summer and time to curl up for a leisurely read in our favourite field, the hobby of computers and electronics and amateur computer programming. No, amateur programming is not dead. It lives on among what are now becoming to be called the classic computers, and even in MS DOS.

And we are back. Rumours of our premature demise have been exaggerated. Indeed, we have been doing some of the exaggerating ourselves. But we're out again with another fine issue with our customary goodies: news, views, address updates and original articles, never seen before this issue's release, such as — page 2, our traditional spot for programming examples, a hardware feature on the last page illustrating how to fix an annoying ZX-81/TS1000 problem, that of loose jacks, and some articles in addition that follow on in our pattern of coverage of the past. An article on terminals and some explanation of technical words involving them is in a way a follow on to our BBS/telephone line communication article on code conversion and BBS file formats, in the last issue. The "Famous Computers" series of articles has explored over the past four years of this newsletter famous computers big and small and their influence on the present microcomputer. Then we added coverage of microprocessors, the invention of which started the whole microcomputer thing rolling. Now, we explore how a computer can win or lose on the market due to its operating system, (O/S), features. Inside we make an examination of the PDP-8, OS/8 operating system, DEC, which to this writer's surprise, was a source of a few of the major features, good and rotten of today's major operating systems including CP/M and MS DOS and even the floundering OS/2 now. Right now when the front page of both daily newspapers and computer periodicals report the battle of the operating systems and how it may break IBM if it doesn't recover gracefully from the OS/2 sales fiasco, Microsoft and IBM falling out over OS/2, Apple and IBM pooling ideas for the next operating system (Apple calls its work 'operation pink'), and in general the looming battle of the titans over operating systems as the key to future microcomputer sales dominance, one can't say it isn't a timely topic. Our background piece on how OS/2 got to where it is and MS DOS, its parent, scoops them all! Yes, we have found a candidate for the source of some of the annoying features of CP/M and MS DOS; what this writer thinks was a mistaken judgment call on the potential power of microcomputers to do multi-tasking at all. We would never say that OS/8 was a bad operating system for its features were much better for its time than MS DOS. It would seem however that the choice of single-tasking for microcomputer operating systems at the very beginning was based on the failure of TSS-8 to deliver and the early success of OS/8 in contrast on the DEC PDP-8. Unfortunately the PDP-8 was far too slow a machine to benchmark operating systems on, compared to even early microprocessors like the 1 MHz 8080. It was not a fair test. But on this experience, today's most popular operating systems were put in a straitjacket from which they are only starting to try to emerge. Example: MS DOS version 5's dual tasking. But there are other reasons for enjoying a sightseeing tour aroubd OS/8. It had some command line features that could still be copied today with advantage, but have not been. Funny how tomorrow's newspaper articles are only understood on doing a bit of operating system (and hardware) archaeology! Join me on page 3 for the digging.

CONTENTS:
-DEC PDP-8 Operating System Had A Hidden Influence on Todays' p.3
-Sinclair Programming Notes— Making A Command Line Interface p.2
-Terminals: From Beginning To End - A Brief Discussion. p.4
-Fixing The Loose Jack Problem on the ZX-81/TS1000 . . . . p.8
- Sinclair News & Views, User Group Addresses & Supplier Update. . . p.7
-ASCII Questions Answered. p.5
-Editorial on Microcomputer Trends And Operating System's Role..above-p.1

☞**IMPORTANT WARNING**: This is a public domain newsletter, privately published, to encourage the use of low cost computers as an educational hobby. Do not hack (modify) your computer hardware, send any money, make any comitments without verifying the info you depend on herein through a fully qualified, local technically and factually reliable personal helper. Computer project help for hobby use only, not commercial application. A little computer bug(☞) appears

Making A Command Line Interface

These days picture-graphic interfaces are having their day, you know, icons and Apple Mac graphics, replacing more and more the older, menu-based interface and maybe to be replaced by voice/animated cartoon user interfaces next. But there will always be a place for the old command line interface as an alternative or as a hacker's entry point one step simpler than machine code fixes. Two uses involve two favourite games of amateur programmers, making an computer system emulator in order to try the command set of a computer or operating system they can't afford normally, and making a little utility program to plug a gap the big companies left in their program line-up. It is easy to see both of these amateur programming subjects need good command line interfaces. They are simpler to write and take less memory; they help the programmer while the program is being written and debugged, and if a more user friendly interface is needed, perhaps it can be added later to your program.

My ZX-81 has an aftermarket disk system that requires a RAND USR to be memorized to SAVE, LOAD or get a directory, so that adding a few lines like:

```
50 PRINT"ENTER SYS COMMAND"
60 INPUT A$
70 IF A$="DISKSAVE" OR A$="DISK" THEN GOTO 9500 :REM disk save is at 9500
80 IF A$="DIRECTORY" OR A$="DIRE" THEN GOTO 9535 :REM directory pull=9535
90 IF A$="" OR A$="EXIT" OR A$=T$ THEN GOTO 1000 :REM 1000=start,T$=prog.
99 GOTO 50                                                        name
```

Actually this is a shortened and cleaned up version of the lines actually used, in the BASIC programs, but they illustrate the use of a non-parsed, simplest sort of command line interface which cuts down on the number of keystrokes and memory work, since GOTO 9535 may not in all programs find the directory routine at that spot. Using a feature of ZX-81 BASIC that not every version has, a line "4 SYS=50" also allows me to get to this command line anytime the program is in RAM (with GOTO SYS).

Command lines are used today in the ARC utility for data compression and in many text editors of the bare-bones kind like MS DOS's EDLIN, so that knowing how to write a command line interface better has many applications for the amateur wanting to get into the utility program writing and/or selling field.

The normal command line uses parsing, the processing of the string that the user enters as a command line, in order to break it up into its fundamental parts and use those parts to extract all the parameters that the program needs to do what the user says is to be done. Note that all parameters should have, arguably, a default value so that if the user does not specify a parameter, a default value is inserted automatically by the program. It may be the last parameter value specified or one that the user specifies when the the program is first used (for example in response to the question, "Do you have a second disk drive, (B:)?", or the program may test the computer's system variables for a second disk drive B:, to see if it exists. Clever use of default features will make the difference between a neat command language and a clumsy one, where the user has to remember not to leave out any parameter and remember all the perhaps odd forms they take. A bad example of a command might be—DIR 1 S W to put a directory of drive#1, on S=screen, in W=wide form (use many columns across the screen). If the user forgets to put in 1, and the routine bombs with an unhelpful, "BAD COMMAND" message, then you have a good example of a user unfriendly interface language. D1, D2, instead of 1,2 may help, good help facilities like displaying the complete command option structure if the user gets stuck as the ZIP utility PKZIP does if no command is used, and more specific error messages, like, "Disk X, is illegal name, choose from 1, 2, 3 ", all would improve it.

In designing the command language, avoid numbers and other cryptic symbols signifying nothing and use in their place mnemonic and evocative symbols with more obvious meaning, their meaning obvious by previous experience to those most likely to be the users of this command line.

In gathering together a parser, subroutines that may be used more than once are useful, chief among them, a lefthand space gobbler, that eliminates irrelevant spaces on the left of the user entered command string. This assumes that spaces on the left have no significance, the normal case, and while spaces are common to use as delimiters, note four things that make their use hazardous : double space delimiters do not mean the same thing as, say, double comma delimiters that can indicate a missing part/parameter that must be replaced by a default value (e.g. ,, =,nothing,) , that they get in by typing error when not intended (commonest typing error), they are never part of a part/parameter (: is part of A:, A-space, space is not part of A) , and printing errors make use of space in user manual hard to see. (Cont. p.7-bot.)

DEC PDP-8 Operating Systems Had A Hidden Influence on Computers Today

What would you say if someone said that the IBM compatible and its MS DOS operating system, the industry standard today, could trace some of its features or lack of them, back to an unprepossessing little computer with only 12 instructions, the DEC PDP-8? Well it was choices made then and experience with the operating systems tried on that little computer a decade before the IBM PC was released, that shaped MS DOS. Perhaps it was the agonizing slowness of some with clock speeds considerably slower than 1 megahertz(IBM PC=4.77 MHz in comparison) in trying to stretch its then precious time via the TSS-8 time sharing operating system, among a number of users, that impressed those who went on to work on microcomputer operating systems, so deeply with frustrating waits at the terminals that did it. But the multi-user type of UNIX clones were eliminated from the list of preferred choices for operating system models when the microprocessor came along. Consequently, not the TSS-8 operating system but the OS/8 operating system ('8' is from the DEC PDP-8's name),was used as the model for CP/M. MS DOS added OS/8-like ideas to its base, the QDOS (QDOS I) system, and so influenced microcomputers up to today. Now, as microcomputers try to shake off the single-user, single-task concept and model themselves on UNIX or the Apple Macintosh concept of windows and stacks of events (not just keyboard but mouse movement and window/ikon placement on the screen) perhaps we should take a look at the PDP-8's OS/8. Let us see how we got here and how we could have taken other directions entirely. We begin with certain innovations and design decisions made in putting together the OS/8 operating system. Today it is show time for new operating systems. Which operating system of those now in contention is likely to be a winner and on the basis of which features will an operating system become a winner in today's world?

As just pointed out, the choice of a single-user, single-tasking type of operating system for virtually all microcomputers up to recently (1991) had maybe been made more on emotion than reason. The middle ground of a single-user, but limited number of tasks, multi-tasking operating system was also eliminated. Ditto the dual-tasking operating system that would allow a computer to run a BBS program or spell check a document or something while still allowing the user the use of the keyboard to peck out the keystrokes for a memorandum or edit the source code of a BASIC program, the sort of thing that DoubleDOS (MS DOS program, about $69) allows. The two-task operating system was certainly known to DEC computing, even if it was more likely to be found in the lab or factory than in Computer Science. It's main use was to allow the scientist or engineer the use of a computer, when the minicomputers were so expensive (tens of thousands of dollars minimum), and still let the same computer control an industrial process or monitor the data from a scientific experiment. Instead of exploring either of these alternatives, which the writer feels is more practical for microcomputers than UNIX with its capacity for an unlimited number of tasks and users, as a replacement for CP/M or MS DOS, for one reason or another, possibly emotion or prejudice, CP/M and in its steps, MS DOS,followed in the patterns set by such early operating systems as OS/8 which channelled all CPU power to one job. Question to answer, in looking down the family tree to the origins of CP/M, MS DOS and a mass of BASIC operating systems used on the older home computers, is, "Are there any peculiarities in OS/8 picked up by other operating systems later that were real winners anyway and any in addition to the narrow focus on single-tasking that were real dogs and trouble us all still?"

The most visible influence of OS/8 on CP/M, MS DOS and maybe UNIX is the CCL. OS/8 Concise Command Language (CCL), was a shorthand to simplify and shorten commands of the language used to enter operating system commands through the keyboard monitor. It included items like DEL, COPY, TYPE, HELP, ED, to mention just the few that have survived in MS DOS/CP/M (the latter two in CP/M only). The CCL feature was an optional set of O/S commands. OS/8 otherwise would run the PDP-8 from longer, more complicated, less intuitive keyboard commands rather than the concise (CCL) ones. The asterisk (*) for wildcards and some file extensions were also adopted by CP/M and MS DOS with small changes, (e.g. .BA to .BAS for BASIC program for example. .EX versus .EXE for another, the executable, compiled program). Unfortunately the copies missed two gems one that let the user use either a full name or a shortened version, like DELETE or DEL, COP or COPY, TY or TYPE, and the other was a standardized set of command options, -L equal to send output to line printer, -S for send to video screen display and so on —— two admirable features neither CP/M nor MS DOS adopted.(OS/8 CCL also had the /letter for options specific to a program like an assembler, so one can't say it had the -options and not the / options)

(continued p6)

Terminals: From Beginning To End - A Brief Discussion

Terminals are the beginning of a computer system where humans key in data and the end of the line for data that the computer will display on the screen or originally would print out on a roll of paper when the old telegraph company keyboard-printers, clunky and noisy, were used in the 1970's. So maybe that is why the word terminal was used, it is a beginning and an end, from the ancient Latin, terminus=boundary. Even though your modern microcomputer may not seem to have a separate terminal unit with keyboard and screen, a lot of the words that are used in the technical manuals to describe keyboard input and output on a video screen are better understood when you see clearly their technical origin in the days of separate terminals, separate from the rest of the computer that is. Anyway, in a time when multi-user and networking systems are coming into the range of capability of microcomputers, we may be hearing much more about these terms. Separate computers may be hooking up even to the old home microcomputer, to get data from them over the telephone lines while we are doing something else with them at our own keyboard for the same computer, at the same time. So a little spade work here with the technical dictionary is probably overdue. Let us begin.

The old clunkamatic teleprinter terminals with their paper rolls were usually used with a roll and ribbon in the machine so that the typist/telegrapher could see while typing so that any typing errors could be picked up. This was a built-in feature of the machine. It would echo the hitting of each key with a character printed on the page, right in front of the typist's eyes. VDU, video display units are similar to that and often have a DIP switch somewhere that can be set to echo to the screen what is going out from the keyboard strokes. And just as with telegraphy, one might wait for a confirmation that the mesaage had been received at the other end, showing that the equipment was turned on and no break in the wire caused it to be lost or garbled. The computer to which the terminal is attached will also usually confirm what it got by echoing it to the screen, that is, displaying it on the video screen. A human on the other end might reword a long text, "Confirm message, your ship sinking, will respond with rescue" that might have been three times as long. Computers aren't so cleverly programmed that they can summarize the message, so to confirm it they simply repeat what they received and send it back along the wires to the user's screen. If the user types, "Hello, Mr. Computer" and the computer echos back on the screen, "XYZVBQCXQYTZ" then it is the intelligence of the human that has to say, "That isn't my message at all; the computer doesn't seem to be getting what I am trying to enter into it." Of course if you have the terminal set to echo the message as it goes out and the computer is echoing what it is receiving, you will get every line double, as in the figure 1.

```
HELLO MR. COMPUTER
HELLO MR. COMPUTER

DIRECTORY
DIRECTORY
```

Figure 1-Double Echoing Bug

Terminals are classified several ways. One is between those "On-site", in your room or building as opposed to "remote terminals" like those calling in on the telephone lines by modem for your BBS. That is fairly simple to understand. Local area network=on-site.

A little more subtle is the "data set" as opposed to "dataterminal". This originated when a computer was attached to a dataset box so that external (outside the computer room, we are talking about the age of big computers now) wires could be connected to the computer and at the other end of these wires, in the building or outside, was a terminal, where the user could work, using the big, central computer. This end of the wire which at that time was connected to a terminal, is called the data terminal side. Since these were usually serial lines rather than parallel that come from the computer, the extra box, the data set usually contained a serial/parallel converter circuit. Now look at it from the computer's eye-view. In a telegraph situation, you will want to have the two stations to have fixed names, if the operators change and are not too smart.

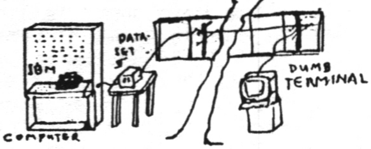

Figure 2 - Terminal-Datasette in "the olden days"

...Continued on page 5...

Terminals (Cont.)

You don't want the operators wasting time with silly conversations over the wires like, "Come in home station, come in home station" and the reply, "You are home station. We are terminal station dummy! Call terminal station not home station." So there was a need for some type of convention here that one end of the line had to be called home or really in this case data set and the other terminal. It doesn't matter which is defined as "home"(data set), but once settled on and the wiring done so that one end is the terminal end and the other end of the wire the home or data set end, the operators stick with it. In this case the operating is mainly done automatically by computer, so that instead of sending, "Come in home station. Are you there down the line", the computer will automatically and without the user's knowledge send an electronic signal that asks in effect, "Data set ready?" and wait for the acknowledgement, "Yes, data set ready". The other side would, the data set would ask, "Terminal ready?" if it wanted to initiate a call, rather than the terminal intiating it by asking if the "data set" on the other end of the line is ready". In order to save on the number of lines needed for this, this sort of protocol is used in wiring the lines, one end is wired as the data set and the other as the terminal, in such a standard as RS-232C. It doesn't matter which is a terminal, a computer, a databox of some kind or just a printer at the end of a serial interface, once you decide which is to be which and wire the RS-232C connectors up properly at each end, standard software can use the RTS=request to send ("Are you there to take a message" and the CTS=clear to send ("Yes I am ready to take your message down; just send it.") to work out if the data terminal is ready (DTR) to receive, everything plugged in and turned on, or the data set is ready (DSR) to receive. Old, manual ways automated, but still showing their relationship to the original big monster computer and teleprinter terminal days and even further back to the telegraph operators one at the home station of the coastguard and the other out on the point at the lighthouse. Obviously somputers are not chatty by nature but brusque. They start up with a beep, not a "Come in home.." TDA (*data set = DCE, terminal = DTE)

Word Confusion: data set as in radio set, not datasette=data cassette, Commodore.

....

What Is An ASCII File and Why Do We Need It?

IBM, a computer pioneer, used BCD, binary coded decimal in its early computers and two BCD numbers could be used to represent a letter of the alphabet in the first of IBM's mass produced computers like the IBM 1620. The IBM 360 of the 1960's extended BCD to include alphabetical characters in a new system, EBCDIC, Extended Binary Coded Decimal Interchange code. In addition, IBM punched cards used a different system, a handful of telegraph printers used different paper tape punching standards, and the situation seem to be becoming anarchic just as minicomputers and the lower cost of big computers were making computers much more numerous. The need for standardization was met by a standard alphanumeric code, (alphanumeric char.="string" characters in BASIC) having 8 bits, one of them saved for parity check redundancy. So with 7-bits to represent characters, 128 codes are possible, 2^7. Well, the IBM PC follows this code closely, just to the lower value 7-bits, and the most significant bit is used not for parity but to add 128 more characters, such things as parts of frames for windows including the straight and corner sections. Different computers can keep close to the ASCII standard by keeping the lower 7-bits of ASCII as the IBM PC does and only using the other bit for any extra 128+ codes for graphics, etc.

Unfortunately although most BBS files and other files that might be exchanged between computers, are for the exchange put into ASCII, say when you hook up to a BBS, by the modem-driving program you are using in your computer, or by excercising an ASCII output option on your wordprocessor or BASIC interpreter SAVE command it is not always an automatic thing. In Commodore 64 BASIC the ASC() command converts a string character into Commodore 64 code, but it is not true ASCII. At least the BASIC in the Timex/Sinclair ZX-81/TS1000 is more honest in calling that command CODE, since it does not use ASCII either. For these computers you will need another computer program to make this conversion.

In practice, ASCII probably means when you hear it used, as a text file with CR*,LF* at the end of each, maximum 252 character line, and no characters with codes higher than 127 (7-bit ASCII) and none lower than 32 (except CR and LF). These are the limits of MS DOS EDLIN, in addition to those of IBM ASCII code, and probably the de facto standards for ASCII in the microcomputer world today.

.... *codes CR=13, LF= 10.

Operating Systems: Out of Sight Is Best?

As of now, summer 1991, the question of which new operating system will be chosen for tommorow's microcomputers, a revised OS/2, UNIX with a shell covering it as in NeXT computer is looming. But is it a trick question? In the future in fact microcomputers as we have known them, big, dumb (unspeaking and unlistening) desktop monsters with less than the power of a supercomputer may become irrelevent. Computerized tasks like wordprocessing may need a control vocabulary of signs but most of what they will do may take little user intervention. Just start typing.

(OS/8 Continued from p.3)

So having seen how admirable the OS/8 was with its command shell language and if anything better rather than poorer than CP/M or MS DOS in that way, we are now in a bit of a quandary. How can we explain the rest of the prominent features of the operating system which are mainly internal and both hard to describe simply and hard to evaluate also, since factors as speed and efficiency are a little hard to measure in an off-the-cuff, qualitative discussion Furthermore it is hard to separate dealing with limitations of the computer,(a 24Kbyte minimum computer in the days when disks held more like 64K rather than 40 megabytes) and attempts to cope with it, from features made possible by and efficiencies achieved by today's superior hardware rather than operating system design advantages.

To mention one OS/8 feature done as well as in MS DOS and CP/M or better take device independent operation. It means programs need not worry about specific I/O device support, such as a wordprocessor that would only work with a certain type of printer unless a special device driver were included with it. This was easier due to proprietary control of DEC over peripherals and the smaller choice. No matter why, OS/8 was more flexible than CP/M or MS DOS in this regard. It certainly was made to handle a variety in memory sizes, VDU's (a DEC video unit,vector scan), TTY (teleprinter of telegraph company fame), big computer tape units, audio cassette style tapes, hard and 8 inch floppy disks, and DEC line-printers. It also had a set of sophisticated utilities, ODT (like DDT & DEBUG machine language editors), a simple text editor that would work even if a screen was not available for echo and cursor movement, and sophisticated text editing with TECO (Text Edit Correction program) that could be called by MUNG system command to enter text into its buffer in one step. ED of CP/M and EDLIN of MS DOS certainly never equalled its features at least in first versions. Some features of TECO are like those in UNIX editors.

But the main influence that OS/8 had (well beyond influencing subsequent DEC operating systems used on say the PDP-11 and VAX 11/780),was that it sold a generation of computer science students (and teachers like Gary Kildall who wrote CP/M) the idea of one-user, single-tasking as the only way to conserve CPU power in the new microcomputers then coming out. Why the worry? Well, in those days even minicomputers were called toys. No computer or O/S could afford to justify such catcalls.

If not with PC's, then I say that a multi-user, multi-tasking system like UNIX does belong with big time-sharing computers like IBM mainframes and Cybers, if for no other reason than time and memory is wasted in being able to support unlimited numbers of users and tasks, a situation that a microcomputer user need not face. But our ignoring the third option has cost us much. Dual or triple-task operating systems or even dual user and triple tasking, would have been a better match for likely uses of micrcomputers in the last decade. Today only networking optimization would have to be added to update the concept and some sort of windowing front-end for new users. A chance missed; today's fast hardware has changed it all.

For home and business computers of the recent past, say of the 32K-512K RAM generation, the combination of automated features and the flexibility in choosing options in the way of tackling certain jobs offered by OS/8 still stands up well as a solid, classic piece of operating system design. In fact, many microcomputer operating systems lack OS/8's depth still today. But today, better graphics and the use of pictures as symbols and signs out of which to build a user command system is the market driven priority. In the heyday of OS/8, memory limitations allowed neither the graphics or operating system program size necessary to go with that approach. But new operating systems still should offer a command line option, and designers should look back at OS/8 as one of the best examples still.

Timex-Sinclair/Cambridge Z-88 News, Views And Contact Addresses (As of July,1991)

It is too bad, but once your group stops sending out a newsletter, other groups just assume that it is dead or dying and you miss a whole lot of information that is circulating in the newsletters. That seems to be the case of the Vancouver Sinclair User Group, which is still alive, but contact address has changed to: VSUG, c/o Mr. Rod Humphrey, 10984 Collings Place, Delta,BC,V4C 7E6....The Ottawa/Hull TSUG on the other hand has folded at the last meeting in February 1991, with former members still working at the hobby in isolation, this newsletter being an example of such activity and produced at my own expense, address: B.J. Harmer,97 Ruskin St.,Ottawa, Ont.,Canada K1Y 4B3 -note address change contemplated in the future-and newsletters are still available as backissues for about $3....Dave Solly, 1545 Alta Vista Dr., Apt.1402, Ottawa,Ont.,Canada K1G 3P4 also still is interested in the TS2068 and in Pascal programming and is hoping to get Sir Clive's Castle BBS started up again in the fall. His home phone is: (613)731-2120....Starting a new North American TS newsletter is a daunting task but Bob Swoger of Chicago and Donald Lambert,1301 Kiblinger Pl.,Auburn,IN,USA 46706 have done just that, disappointed in the lack of newsltr. from SNUG, the older group. Contact Don if you have an article to submit....Don is also working on an Unofficial Guide to the Aerco TS2068 Disk Interface....A new contact at least for me is the Dayton Microcomputer Association, c/o Mr. Gary Ganger, 812 Hedwick St.,New Carlisle,Ohio,USA45344-2619-Gary is the TSUG sub-group contact.. ....coverage of TS matters continues in the newsstand magazine Computer Monthly, which continues to run in each article lists of Sinclair product suppliers. -- This magazine also handles CP/M items...LIST (New York City) has a new newsletter editor, contact-5 Peri Lane,Valley Stream,NY,USA11581 for subscription-they also have been publishing their own books of tips on the ZX-81....and so have we with the Tips on the ZX-81 book of our own running to 50 pages now including such stuff as on p.8 of this newsletter which is taken from our book-our price$20, since the writer does not have a copy of LIST's book it would not be fair to try to review it but our readers should know that support for the ZX-81 continues in unexpected ways such as these....Newsletters received recently from:CATUG,c/o Bob Swoger,613 Parkside Circle, Streamwood,IL,USA60107..Toronto,c/o Geo.Chambers,14 Richome Ct.,Scarborough,Ont., Canada M1K 2Y1..SMUG,P.O.B.101,Butler,WI,USA53007..in addition to others mentioned above...**QL** NEWS: International QL Report,4-6 per yr.,US$10,QL accessory news,from Seacoast Services,15 Kilburn Court,Newport,RI,USA 02840....QL Survivors Source Book lists all known QL suppliers internationally, US$7, from TS magazine, UPDATE,POB 1095,Peru,IN,USA46970....QL Sub mag from Britain is gone...Senior Net(55+)has BBS's

The whole question of where hobby computing is going is coming up these days; Where have all the programmers gone? Do new users learn BASIC anymore? Probably not. Are older makes and models junk or antique collectibles in a period of transition? If they really are in the 'classic' period of transition towards antiques will anyone but the remaining user group people know how to use or fix them? If most small, make-specific user groups dwindle or vanish should there be bigger, non-make specific, classic computer user groups formed? (Weren't Peirce-Arrow clubs replaced by antique car clubs, not make-specific?) As the IBM PC becomes the hobby computer of choice for most, will its more expensive replacements with very complicated hardware and software at the nuts and bolts level open a niche soon in the market for a simpler, hardware hackable, less expensive hobby computer? Until it does we shall just keep using our orphaned classic computers!....New electronic devices in the news: microminiaturized ferrite memories, vacuum tubes, and now IBM has electron-sized nanominiaturization which may be used first for resistor and capacitor miniaturized substitutes....Our address for this newsletter, publisher— Bill Harmer,97 Ruskin St.,Ottawa,Ont.,Canada K1Y 4B3—drop us a line, (letter or disk). Publications available approx. 40¢ per page: ZX-81 Ass.Lang.(32),ZX-81 LDOS(120pages) ZX-81 Tips & Tricks(50 pages), Generic Orphan Booklet (35 pages),Generic Disk Guide(")

(Continued from page 2) Command Line Interfaces -Programmer's View...

After defaults, mnemonics, built-in help facilities, and a language design that makes errors stand out so the program can catch them with simple routines comes a shrewd estimate of the users' background and previous knowledge. Cryptic command lines may work in a programmer's editor if it has the same syntax of a language, programming or operating systems command language, that the programmers already use. If it is aimed at office typists it should emulate the look and feel of their typewriter — a blank piece of paper and special function keys along the bottom.

Command lines are not dead but lack of forethought can make yours deadmeat.

……

ZX-81/TS1000 Loose Jack Problem-The Fix

The three jacks on the side of this computer that accept mic., ear and power plugs are of a somewhat flimsy construction that may become apparent after a few years of use (or earlier on rough usage) of the computer. The cure is fortunately rather simple for the mechanically/electronically handy user or technician.

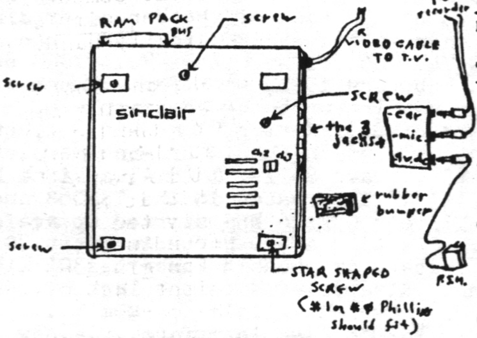

Fig.1 -ZX-81/TS1000 Bottom View
(the 5 screws to remove)

The Fix: Step 1 - take the bottom half of the case off (without removing the printed circuit board inside or the keyboard membrane plastic sheet at this time). There are 5 screws that must be removed from the bottom of the case. This is not as easy as it might seem since only two are visible. The other three are found under the rubber feet (rectangular strips glued on near the corner). These can simply be pulled off in order to remove the screws under them. Once the screws are completely removed, the bottom half of the plastic case should separate with only light prying. Any force indicates that the screws are not completely removed. See Fig. 1.

Step 2 - The printed circuit board is now secured to the top half of the plastic case by two screws. See where they are (upper to mid-, left near the RAM pack opening and near the centre, more towards the middle of the case, typically). For reassembly, note where they came from, with a soft pen say).. The printed circuit board must be moved up and away from the top half of the case about one or two inches only (so as not to disconnect or damage the fragile keyboard matrix/membrane plastic from the connector on the board)- Fig.2.

Step 3- holding the printed circuit board so that one can see the three little plastic boxes that house the jacks, any prongs that are bent too far up and out, may be carefully pushed back into position with a small screwdriver or toothpick. Several pushes back may be necessary until the springiness is overcome in its attempt to reposition the prong outwards. Now test the jacks with the cable for the cassette recorder (no power applied) for mechanical fit.

Step 4 - Now you are ready to reassemble the computer making sure that the two screws for the printed circuit board are not inserted in the holes meant for the outside, bottom shell of the case, screws. If you put one or both of these screws in the wrong place, the screws will not all go in when you try to screw the back on again and all the screws on the outside will have to be removed and the bottom taken off again to find the source of the problem. Do not disconnect the little metal bar or whatever, that ground contact with the metallic paint inside the case that provides some measure of protection from RFI for the t.v. etc.

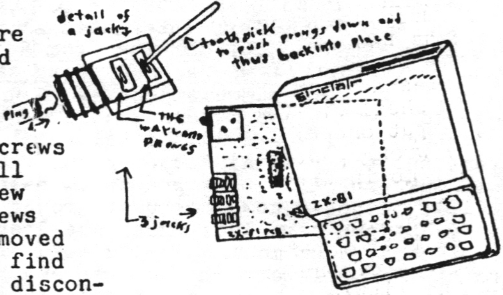

Fig.2-The Three Jacks Inside, On the Circuit Board

Step 5 - Once the printed circuit board is secured inside the case with its two screws and the bottom of the case secured with its five screws and the little rubber feet or runners stuck back on, you are ready to test the computer to see if the original problem is fixed (and no new ones added).

Note that loose power jack into the computer can also cause the save/load crashes and some users have either tightened up that jack too, or replaced it with hardwiring the power supply wire to the p.c.b. and adding some safety device like a switch on the line or pilot light (standard LED and ½ watt, 1K ohm for example) to make sure it is not left on.

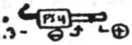

Fig.3- DON'T REVERSE POLARITY!